LISA สูตรลับการเขียนภาษาแอสเซมบลีให้เป็นของง่าย

ณรงค์พร เหล่าศรีสิน

ท่านผู้อ่านที่ติดตามบทความเรื่องการเขียนโปรแกรมภาษาแอสเซมบลีมาตั้งแต่แรก จนถึงเดี๋ยวนี้ก็หลายตอนแล้ว คงจะเกิดความรู้สึกอยากเขียนโปรแกรมแอสเซมบลีในสไตล์ของตัวเองบ้าง หลังจากที่คีย์โปรแกรมตามตัวอย่างในบทความมาหลายครั้งแล้ว ซึ่งถ้าหากคุณเห็นด้วยกับผมล่ะก็ ร่างโปรแกรมของคุณเตรียมไว้ได้เลย เราจะมาเขียนโปรแกรมภาษาแอสเซมบลีกัน

วิธีการเขียนโปรแกรมภาษาแอส–เซมบลีเข้าสู่เครื่องคอมพิวเตอร์นั้น เดิมทีเราจะต้องเปลี่ยนภาษาแอสเซมบลีไปเป็นภาษาเครื่องซึ่งมีลักษณะเป็นตัวเลขฐานสิบหกทั้งหมด หลังจากนั้นจึงป้อนรหัสตัวเลขเหล่านี้ลงในหน่วยความจำ โดยผ่านทางโปรแกรมมอนิเตอร์ในเครื่องคอมพิวเตอร์แอปเปิ้ลทู การป้อนโปรแกรมในลักษณะนี้ เท่าที่ผ่านมาก็ดูเหมือนจะไม่มีความยากลำบากมากนัก เพราะคุณเพียงแต่ป้อนตัวเลขเข้าไปในเครื่องตามโปรแกรมตัวอย่างเท่านั้นก็สามารถรันโปรแกรมได้แล้ว แต่สำหรับตอนนี้คุณกำลังคิดจะหัดเขียนโปรแกรมด้วยตัวคุณเองอยู่นะ ดังนั้นหน้าที่ต่าง ๆ จึงต้องตกเป็นของคุณเพียงคนเดียว คือ เริ่มตั้งแต่การเขียนโปรแกรมภาษาแอสเซมบลี แปลงไปเป็นรหัสภาษาเครื่อง แล้วจึงป้อนเข้าเครื่องคอมพิวเตอร์ แต่เท่านี้ยังไม่พอหรอก เพราะถ้าหากโปรแกรมของคุณมีข้อผิดพลาดเกิดขึ้นแล้ว คุณก็ต้องมีหน้าที่ตรวจสอบและแก้ไขโปรแกรมเพิ่มอีกด้วย

อ่านไป นึกภาพไป คุณก็คงเห็นว่ามันช่างเป็นงานที่ลำบากลำบนอะไรเช่นนี้ อย่าเขียนเลยดีกว่า เจ้าโปรแกรมแอสเซมบลี นี่ถ้าหากปล่อยเอาไว้อย่างนี้ก็คงจะไม่มีใครอยากเขียนโปรแกรมแอสเซมบลีแน่ ๆ ดังนั้นจึงได้มีการคิดค้นหาเครื่องมือชนิดหนึ่งขึ้นมา เพื่ออำนวยความสะดวกแก่ผู้ใช้คอมพิวเตอร์ทั้งหลาย เครื่องมือดังกล่าวนี้มีลักษณะเป็นโปรแกรมหรือซอฟท์แวร์ ดังนั้นเราจึงเรียกเครื่องมือนี้ว่า software tool

ตัวซอฟท์แวร์ทูลนั้น มีด้วยกันหลายแบบหลายโปรแกรม ตามแต่ว่าจะนำไปช่วยงานด้านใด ตัวอย่างเช่น เทอร์โบ ปาสคาล, ไซด์คิด, โลตัส เป็นต้น แต่สำหรับ LISA นี้เป็นซอฟท์แวร์ทูลที่ใช้ช่วยในการเขียนโปรแกรมแอสเซมบลีของซีพียู 6502 ช่วยให้คุณสามารถจะเขียนโปรแกรม, แก้ไขโปรแกรม เพิ่มเติมหรือตัดต่อโปรแกรมได้โดยง่าย และโดยเฉพาะอย่างยิ่งช่วยแปลภาษาแอสเซมบลีไปเป็นภาษาเครื่องด้วย

LISA เป็นชื่อย่อของชื่อเต็มที่ว่า Lazer's Interactive Symbolic Assembler ชุดโปรแกรมของ LISA นั้น จะประกอบด้วยโปรแกรมต่าง ๆ ดังรูปที่ 1

เมื่อคุณมีชุดโปรแกรมลิซ่าอยู่ในแผ่นดิสค์แล้วก็เริ่มรันโปรแกรมได้เลย

1

```
A   002  HELLO
B   002  LISA V2.5
B   058  ASM.CODE
B   050  ASM.CODE.1
B   016  ASM.CODE.2
B   037  MPTOLISA.CODE
L   050  RANDYS HIRES ROUTINES
L   044  HIRES.2
L   039  DIASM.TXT
L   038  DISASM.2
L   026  SCTOLISA.TXT
B   003  SORT
B   006  SCTOLISA
T   004  LISA
```

RUN HELLO หรือ BRUN LISA V2.5 เพื่อเริ่มรันโปรแกรมลิซ่า

เป็นโปรแกรมภาษาแอสเซมบลี ที่เขียนขึ้นโดยลิซ่า สังเกตจากชนิดของไฟล์ที่ลิซ่าเปลี่ยนจาก B เป็น L

จะได้ยินเสียงดังจากดิสค์ไดรฟ์ รอสักประเดี๋ยวหนึ่ง พอเรียบร้อยแล้วก็จะพบข้อความทักทายดังรูปที่ 2

คุณคงสังเกตเห็นเครื่องหมาย "!" ซึ่งเป็นพรอมท์และมีเคอร์เซอร์กะพริบอยู่ข้างล่างแล้ว นี่เป็นลักษณะที่แสดงว่าขณะนี้ลิซ่ากำลังเตรียมพร้อมรับคำสั่งจากคุณอยู่ ดังนั้น การเรียนรู้ลิซ่าในขั้นแรกนี้ก็คือ คุณควรรู้จักกับคำสั่งต่าง ๆ ของลิซ่าเสียก่อน โดยดูจากรูปที่ 3

คำสั่งต่าง ๆ ที่แสดงให้ดูนั้นมีลักษณะพิเศษอย่างหนึ่งคือ เมื่อเวลาใช้งานเราไม่จำเป็นต้องป้อนคำสั่งนั้นเต็มคำก็ได้ เพียงแต่ป้อนเฉพาะตัวอักษรที่อยู่นอกวงเล็บก็พอ ตัวอย่างเช่น เมื่อคุณต้องการใช้คำสั่ง INSERT คุณก็อาจป้อนแค่ตัวอักษร I เท่านั้น เครื่องก็สามารถเข้าใจได้แล้ว

ทีนี้เราลองมาดูกันซิว่า คำสั่งเหล่านี้ใช้ทำอะไรได้บ้าง

I(NSERT) เป็นคำสั่งสำหรับแทรกโปรแกรม เราสามารถใช้ในตอนเริ่มเขียนโปรแกรม หรือจะใช้เมื่อต้องการแก้ไขโปรแกรมก็ได้

เมื่อเราต้องการเริ่มเขียนโปรแกรมนั้น เราเพียงแต่ป้อนว่า I และกด return เครื่องก็จะแสดงตัวเลข 1 ออกมา และมีเคอร์เซอร์กะพริบอยู่ข้าง ๆ ตอนนี้คุณก็สามารถป้อนโปรแกรมได้แล้ว

ตัวเลข 1 ที่ปรากฏอยู่ข้างหน้านั้นเราจะเรียกว่า LINE NUMBER ก็คงจะได้ เช่นเดียวกับในภาษาเบสิก ตัวเลขเหล่านี้เป็นตัวเลขที่ลิซ่ากำหนดให้เองจะเพิ่มขึ้นเรื่อย ๆ เมื่อจำนวนบรรทัดของโปรแกรมเพิ่มขึ้น ตัวเลขบอกบรรทัดนี้มีประโยชน์ในการอ้างอิงตำแหน่งของโปรแกรม ดังเช่นว่า เราต้องการจะแทรกโปรแกรมบางส่วนไว้ข้างหน้าโปรแกรมเดิมที่บรรทัดที่ 7 เราก็ป้อนว่า

```
! return
```

ที่นี้เครื่องก็พร้อมจะรับโปรแกรมจากคุณให้เข้ามาแทรกหน้าบรรทัดที่ 7 แล้ว คุณสามารถป้อนโปรแกรมอีกกี่บรรทัดก็ได้เท่าที่คุณต้องการ จนเมื่อคุณต้องการหยุดก็ทำได้โดยกด ctrl-E และ return ลิซ่าก็จะออกจากโหมดแทรกโปรแกรมและแสดงพรอมท์(!) ขึ้นมา

สรุปแล้ว การใช้คำสั่งแทรกโปรแกรมมีอยู่ 2 ลักษณะ คือ

1. แทรกโดยไม่กำหนด LINE NUMBER ไว้ ลิซ่าก็จะเข้าสู่โหมดแทรกโปรแกรมที่บรรทัดสุดท้ายของโปรแกรม ใช้ในกรณีที่คุณต้องการเขียนโปรแกรมเพิ่มเติมต่อท้ายอีก แต่ถ้าหากคุณใช้คำสั่งแทรกหลังคำสั่ง NEW หรือไม่มีโปรแกรมภาษาแอสเซมบลีอยู่ในเครื่องเลย ลิซ่าก็จะถือว่าเป็นการแทรกโปรแกรม หรือเริ่มป้อนโปรแกรมที่บรรทัดที่หนึ่งทันที

2. กำหนดบรรทัดที่ต้องการแทรก ใช้เมื่อเราต้องการแทรกโปรแกรมไว้ก่อนหน้าโปรแกรมเดิมตรงตำแหน่งบรรทัดที่กำหนด

D(ELETE) เป็นคำสั่งในการลบบางส่วนของโปรแกรมออกไป ซึ่งอาจจะลบเพียงบรรทัดเดียวหรืออาจจะลบหลาย ๆ บรรทัดเลยก็ได้ ดังนี้

DELETE 5 ลบโปรแกรมเฉพาะ บรรทัดที่ 5

DELETE 4, 10 ลบโปรแกรมตั้งแต่บรรทัดที่ 4 จนถึงบรรทัดที่ 10

2

```
LAZER SYSTEM'S
INTERACTIVE
SYMBOLIC
ASSEMBLER

(C) 1981 LAZER SYSTEMS
    VERSION 2.5D -- 48 K
```

จำนวนหน่วยความจำแรมในเครื่อง

3

```
N(EW)     LO(AD)     SA(VE)     CTRL-D
L(IST)    I(NSERT)   D(ELETE)   CTRL-P
A(SM)     AP(PEND)   LE(NGTH)
BRK       M(ODIFY)
```

เมื่อมีการเปลี่ยนแปลงโปรแกรม ไม่ว่าจะโดยการลบโปรแกรม, การต่อเติม หรือการแทรกโปรแกรม โดยใช้คำสั่งประเภท DELETE, INSERT, MODIFY ก็จะทำให้มีการเปลี่ยนแปลงตัวเลขประจำบรรทัดโดยอัตโนมัติ เช่น เมื่อใช้คำสั่งลบโปรแกรมบรรทัดที่ 4 ถึงบรรทัด 10 ก็จะทำให้โปรแกรมในช่วงนั้นหายไป จากนั้นโปรแกรมเดิมที่เคยอยู่บรรทัดที่ 11 ก็จะถูกเลื่อนให้มาอยู่บรรทัดที่ 4 แทน โปรแกรมที่อยู่บรรทัดถัดไป ก็จะเลื่อนขึ้นมาแทนส่วนที่ว่างนี้เช่นกัน หรือในกรณีที่คุณใช้คำสั่งแทรกโปรแกรมที่บรรทัดที่ 7 โปรแกรมใหม่ที่ได้รับเข้ามาก็จะอยู่ในบรรทัดที่ 7 ส่วนโปรแกรมเดิมในบรรทัดที่ 7 และบรรทัดถัดไปก็จะถูกเลื่อนลงไปเรื่อย ๆ จนกว่าจะออกจากโหมดแทรกโปรแกรม จากตัวอย่างที่ยกมาอธิบายนี้ ก็คงจะเห็นแล้วว่าตัวเลขประจำบรรทัดนี้จะมีการเปลี่ยนแปลงอยู่เสมอ ดังนั้นก่อนที่เราจะทำการเปลี่ยนแปลงอะไรในโปรแกรม ก็ควรจะตรวจดูตัวเลขบรรทัดให้ถูกต้องเสียก่อน โดยการใช้คำสั่ง LIST ช่วยแสดงโปรแกรมออกมา ซึ่งผมกำลังจะกล่าวถึงในตอนถัดไปนี้แหละ

L(IST) เราสามารถใช้คำสั่งนี้เพื่อดูโปรแกรมภาษาแอสเซมบลีที่เราเขียนขึ้นได้ โดยมีวิธีใช้หลายแบบดังนี้

LIST	จะแสดงโปรแกรมทั้งหมดจนจบ
LIST 2	จะแสดงเฉพาะโปรแกรมบรรทัดที่ 2
LIST 2, 7	แสดงโปรแกรมตั้งแต่บรรทัดที่ 2 ถึงบรรทัดที่ 7
LIST 2,	แสดงโปรแกรมตั้งแต่บรรทัดที่ 2 และบรรทัดถัดมา จำนวน 22 บรรทัด

ในบางขณะที่เครื่องกำลังแสดงโปรแกรมอยู่นั้น หากคุณต้องการจะหยุดลิสต์ ก็สามารถทำได้ 2 วิธี คือ

– กด ctrl-C เพื่อเบรคลิซ่าเอาไว้ ด้วยวิธีนี้จะทำให้ลิซ่าออกจากโหมดแสดงโปรแกรม ขึ้นเครื่องหมาย ! และรอรับคำสั่งใหม่จากเราต่อไป

– กด SPACE BAR เพื่อหยุดลิสต์โปรแกรมชั่วคราว และเมื่อต้องการให้แสดงโปรแกรมต่อไปอีก ก็ให้กดปุ่มใดก็ได้อีกครั้ง

บางครั้งคุณอาจจะติดนิสัยบางอย่างจากการใช้งานภาษาเบสิกมาก ๆ เช่น การใช้ปุ่ม ctrl-S เพื่อหยุดลิสต์โปรแกรมชั่วขณะ แต่กับลิซ่านั้น บางครั้งปุ่ม ctrl-S ก็อาจจะใช้ไม่ได้ผลดีนัก เพราะเกิดอาการลักปิดลักเปิด หรือกดแล้วก็ไม่ยอมหยุดเอาดื้อ ๆ ต้องกดกันหลาย ๆ ครั้งกว่าจะหยุดได้

M(ODIFY) ใช้สำหรับแก้ไขโปรแกรมในบรรทัดใดบรรทัดหนึ่ง โดยการใช้คำสั่งนี้จะเหมือนกับการใช้งานคำสั่ง 3 คำสั่งตามลำดับคือ LIST, DELETE, INSERT ดังนั้นเมื่อใช้คำสั่งแก้ไขโปรแกรมตามตัวอย่างในรูปที่ 4 ก็จะทำให้เกิด .....

จากนี้คุณก็สามารถแก้ไขโปรแกรมในบรรทัดที่ 7 ได้แล้ว โดยการพิมพ์ข้อความใหม่ลงไปเลย หรือจะเคลื่อนย้ายเคอร์เซอร์ไปยังตำแหน่งที่ต้องการแก้ไขก็ได้

เช่นว่า คุณต้องการเปลี่ยน LDA #$30 ในบรรทัดที่ 7 ไปเป็น LDA#$50 ก็ให้ใช้คีย์ ctrl-O เลื่อนเคอร์เซอร์ขึ้นไปข้างบนบรรทัด แล้วจึงใช้ปุ่ม→ เลื่อนไปที่เลข 3 เปลี่ยนเป็นเลข 5 จากนั้นเลื่อนเคอร์เซอร์ด้วยปุ่ม→ จนสุดบรรทัด

4

! M 7	ใช้คำสั่งแก้ไขโปรแกรมบรรทัดที่ 7
7 LDA #$30	แสดงโปรแกรมในบรรทัดที่ 7
7 cursor	ลบโปรแกรมในบรรทัดที่ 7 แล้วเข้าสู่โหมดแทรกโปรแกรมในบรรทัดที่ 7

แล้วตามด้วย return ก็เป็นอันเสร็จการแก้ไข

นอกจากการใช้คำสั่ง MODIFY ในลักษณะดังกล่าวแล้ว เราอาจใช้คำสั่งนี้แก้ไขโปรแกรมทีละหลาย ๆ บรรทัดก็ได้ เช่น

ตัวอย่างในรูปที่ 5 เป็นวิธีแก้ไขโปรแกรมทีละหลาย ๆ บรรทัด มีหลักการเช่นเดียวกับที่กล่าวมาแล้วข้างต้น คือใช้ปุ่ม ctrl-O เลื่อนเคอร์เซอร์จากบรรทัดที่ 37 ② ขึ้นไปยังบรรทัดที่ 37 ① จากนั้นก็เริ่มแก้ไขโปรแกรมตั้งแต่บรรทัดที่ 37, 38,... จนจบบรรทัดที่ 40

เนื่องจากตอนนี้ลิซ่ากำลังอยู่ในโหมดแทรกโปรแกรม ดังนั้นถ้าหากต้องการแทรกโปรแกรมเพิ่มเติมลงไปก็สามารถทำได้เลย หรือจะออกจากโหมดแทรกโปรแกรม โดยกดปุ่ม ctrl-E เลยก็ได้

N(EW) เป็นคำสั่งที่มีผลเช่นเดียวกับในภาษาเบสิกคือ ใช้ลบโปรแกรมที่มีอยู่ในเครื่องและเริ่มต้นรับโปรแกรมใหม่ โปรแกรมที่ถูกลบนี้เป็นเฉพาะส่วนภาษาแอสเซมบลีเท่านั้น แต่ไม่สามารถลบโปรแกรมภาษาเครื่องที่ผ่านการเปลี่ยนมาแล้ว

เมื่อมีการใช้คำสั่ง NEW นี้ ลิซ่าก็จะขึ้นเครื่องหมายคำถาม พร้อมทั้งถามว่าจะให้ลบโปรแกรมทั้งหมดแน่หรือ ซึ่งถ้าหากคุณยังยืนยันที่จะลบ ก็กดปุ่ม Y ไป แต่ถ้าคุณเกิดลังเลใจขึ้นมาก็สามารถยกเลิกคำสั่ง NEW ได้โดยตอบลิซ่าไปด้วยปุ่ม N

LO(AD) และ SA(VE) หลังคำสั่งทั้งสองนี้จะตามด้วยชื่อของไฟล์ที่เป็นโปรแกรมภาษาแอสเซมบลีซึ่งมีลักษณะเป็นตัวอักษรหรือเก็บอยู่ในรูปของรหัส ASCII การใช้คำสั่ง SAVE หรือ LOAD นี้ แท้ที่จริงแล้วก็คือ การใช้คำสั่ง BSAVE หรือ BLOAD นั่นเอง แต่ลิซ่าทำเอาไว้พิเศษนิดนึงตรงที่โปรแกรมซึ่ง SAVE ด้วยลิซ่านี้ จะมีรหัสบ่งชนิดของไฟล์เป็นตัว L แทนที่จะเป็นตัว B(INARY) ซึ่งคุณสามารถใช้คำสั่ง CATALOG เอาชื่อไฟล์ออกมาดูได้ เมื่อยังอยู่ในโปรแกรมลิซ่า แต่เมื่ออยู่นอกโปรแกรมลิซ่าแล้ว ตัวอักษร L จะเปลี่ยนเป็นตัวอักษร B ดังเดิม

AP(PEND) เป็นคำสั่งสำหรับนำโปรแกรมภาษาแอสเซมบลีจากแผ่นดิสก์เข้าสู่เครื่อง ดังนั้นจึงมีลักษณะคล้ายกับคำสั่ง LOAD ที่ผ่านมา จะต่างกันก็ตรงที่ เมื่อเราใช้คำสั่งโหลดแล้ว โปรแกรมใหม่จากแผ่นดิสก์จะเข้ามาทับโปรแกรมเก่าที่มีในเครื่องให้หายไป แต่สำหรับคำสั่ง APPEND นั้น จะนำโปรแกรมใหม่มาต่อท้ายโปรแกรมเดิมที่มีอยู่ในเครื่อง

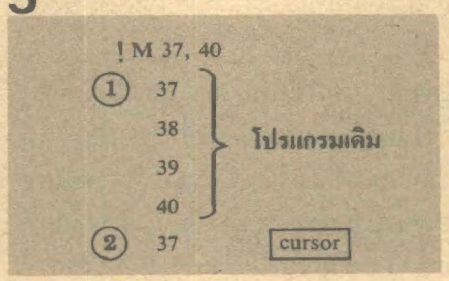
A(SM) เราใช้คำสั่ง ASM เมื่อเราต้องการให้ลิซ่าเปลี่ยนโปรแกรมภาษาแอสเซมบลีไปเป็นโปรแกรมภาษาเครื่องตามที่เราต้องการ บางครั้งในขณะที่ลิซ่ากำลังแปลภาษาอยู่นั้น ถ้าหากมีการผิดพลาดเกิดขึ้น เครื่องก็จะฟ้องเราว่า โปรแกรมภาษาแอสเซมบลีนั้นมีการผิดพลาดเพราะเหตุใด พร้อมกับถามเราว่า

C (ONTINUE) OR A(BORT) ?

ถ้าคุณกดปุ่ม C เครื่องก็จะทำการแปลโปรแกรมของคุณต่อไป แต่ถ้าหากคุณกดปุ่ม A เครื่องก็จะออกจากโหมดแปลภาษาพร้อมกับรอรับคำสั่งต่อไปจากคุณ ซึ่งขณะนั้นคุณก็อาจออกคำสั่งให้แก้ไขข้อผิดพลาดของโปรแกรมนั้นก็ได้

BRK บ่อยครั้งที่เมื่อคุณแอสเซมเบิลโปรแกรมเป็นภาษาเครื่องเรียบร้อยแล้ว คุณเกิดอยากจะทดลองรันโปรแกรมนั้นขึ้นมาทันที คุณสามารถใช้คำสั่ง BRK นี้เพื่อออกจากลิซ่าแล้วเข้าสู่โปรแกรม

5



มอนิเตอร์ของเครื่องแอปเปิ้ลได้เลย เมื่อเป็นดังนี้แล้ว คุณก็สามารถใช้คำสั่ง G เพื่อลองรันโปรแกรมได้

หลังจากที่คุณได้ทดลองโปรแกรมภาษาเครื่องนั้นแล้ว เกิดอยากจะกลับเข้าสู่ลิซ่าใหม่อีกครั้งก็สามารถทำได้หลายวิธี เช่น ป้อนประโยค 3D0G หรือ 6003G(กรณีที่แอปเปิ้ลมี RAM 48K) หรือใช้ E003G (เมื่อมี RAM ครบ 64K) เป็นต้น สำหรับการเข้าสู่ลิซ่าใหม่นั้นยังมีรายละเอียดอีกเล็กน้อยที่ควรจะทราบ แต่ผมจะขอเก็บเอาไว้ก่อน จะกล่าวถึงอีกครั้งหนึ่งในตอนหลังนะครับ

LE(NGTH) หลังจากใช้คำสั่งนี้แล้ว เครื่องจะพิมพ์ค่าความยาวของโปรแกรมภาษาแอสเซมบลีที่มีอยู่ในเครื่องตอนนั้น (ไม่ใช่ความยาวของโปรแกรมภาษาเครื่อง) เราจะใช้ประโยชน์จากคำสั่งนี้ได้เมื่อตอนที่เราเขียนโปรแกรมภาษาแอสเซมบลีที่ยาวมาก ๆ หรือนำโปร–แกรมหลาย ๆ ส่วนมาต่อกันด้วยคำสั่ง APPEND เราควรจะตรวจสอบดูอยู่เสมอว่า โปรแกรมของเรานั้นยาวเกินขอบเขตที่ลิซ่ากำหนดไว้หรือไม่ เพราะถ้าหากว่าโปรแกรมนั้นยาวเกินไป ส่วนที่เกินออกไปนี้ ก็จะไปทำลายพื้นที่หน่วยความจำส่วนอื่นของลิซ่าที่จัดไว้ให้เสียหายได้

สำหรับค่าความยาวของโปรแกรมภาษาแอสเซมบลีที่สามารถบรรจุในเครื่องได้นั้น ขึ้นอยู่กับปริมาณหน่วยความจำในเครื่องคอมพิวเตอร์ ถ้าเครื่องของคุณมีแรมอยู่ 48K ก็จะเก็บโปรแกรมได้ยาว $ 4000 ไบท์ ในขณะที่แรม 64K สามารถเก็บโปรแกรมได้ยาวขนาด $6800 ไบท์

ctrl-P เป็นปุ่มที่สามารถบังคับให้ลิซ่ากระโดดไปยังแอดเดรสที่ $E009 ได้ ซึ่ง ณ แอดเดรสนี้จะมีคำสั่งกระโดดไปยังโปรแกรมอื่น ๆ ที่อยู่ในเครื่องอีกทีหนึ่ง สำหรับพื้นที่หน่วยความจำที่ใช้เก็บโปรแกรมนั้น อาจจะเก็บไว้ที่ $9480-95FF ได้

ctrl-D เพื่อเป็นการเพิ่มประสิทธิภาพให้กับลิซ่า ผู้ออกแบบจึงไม่เพียงแต่ทำให้ลิซ่ามีคำสั่งต่าง ๆ ของตัวเองเท่านั้น แต่ยังทำให้สามารถใช้คำสั่งต่าง ๆ ที่มีอยู่ในแอปเปิ้ลซอฟท์เบสิกด้วย เมื่อเราต้องการใช้คำสั่งใดก็ให้กดปุ่ม ctrl-D นำหน้าก่อนแล้วจึงคีย์คำสั่งตามลงไป

ชุดคำสั่งที่สามารถตามท้าย ctrl-D ได้แก่

1. PR#n และ IN#n เมื่อ n คือหมายเลขสล้อตจาก 0 ถึง 7

2. คำสั่งต่าง ๆ ในดอส เช่น CATALOG, BSAVE, BLOAD เป็นต้น

เพียงการใช้ปุ่ม ctrl-D นี้ ทำให้ลิซ่ามีความสามารถเพิ่มมากขึ้นอีกหลายเท่า ที่นี้คุณก็จะสามารถสั่งให้ลิซ่าแสดงโปรแกรมออกมาทางเครื่องพิมพ์ หรือจะขอดูชื่อไฟล์ต่าง ๆ ในแผ่นดิสค์ก็ได้ หรือจะเก็บโปรแกรมภาษาเครื่องที่ผ่านการแปลมาแล้วลงบนแผ่นดิสค์ได้เช่นกัน และ...อื่น ๆ อีกมากมาย

เป็นอย่างไรบ้างครับ สำหรับการใช้งานคำสั่งต่าง ๆ ของลิซ่าที่ผมอธิบายให้ฟังนี้ ผมอยากจะแนะนำให้คุณได้ทดลองใช้ดู แล้วคุณจะพบว่าคุณสามารถสั่งงานลิซ่าได้ง่ายและคล่องแคล่วในเวลาอันสั้น สำหรับตอนนี้เรื่องถัดไปที่จะมาพูดถึงกันก็เห็นจะเป็นเรื่อง โครงสร้างของภาษาแอสเซมบลีและเรื่องของคำสั่งเทียม ซึ่งอาจจะต้องคุยกันยาวหน่อย แต่พอหลังจากนั้นแล้ว คุณก็คงสามารถพอที่จะสนุกกับภาษาแอสเซมบลีได้บ้างแล้วหละ!

โครงสร้างของภาษาแอสเซมบลี

ในโปรแกรมภาษาแอสเซมบลีนั้นจะแบ่งแยกออกเป็นส่วนต่าง ๆ ที่สำคัญอยู่ 4 ส่วนคือ

1. ส่วนของเลเบล(LABEL)

2. ส่วนของนีโมนิค(MNEMONIC)
3. ส่วนของโอเปอร์แรนด์(OPERAND)
4. ส่วนของคำอธิบาย(COMMENT)

ในประโยคคำสั่งแต่ละอันนั้น อาจจะประกอบด้วยส่วนต่าง ๆ ข้างต้นเพียงบางส่วนก็ได้ ไม่จำเป็นว่าจะต้องมีให้ครบทั้ง 4 ส่วน แต่ถ้าในหนึ่งประโยคมีหลาย ๆ ส่วนรวมกัน เราจะแบ่งแต่ละส่วนนี้ออกจากกันโดยใช้ช่องว่างคั่นส่วนต่าง ๆ เอาไว้ เราลองมาดูรายละเอียดของส่วนประกอบเหล่านี้กันสักหน่อยดีกว่า

เลเบล เป็นข้อความที่เราใช้อ้างอิงแทนค่าแอดเดรส คือเราจะไม่กำหนดค่าแอดเดรสจริง ๆ แต่จะใช้เลเบลในลักษณะเป็นตัวแปรตัวหนึ่งที่ใช้เก็บค่าแอดเดรสแทน ดังนั้นเวลาเราเขียนโปรแกรมจึงไม่จำเป็นต้องพะวงถึงค่าแอดเดรสจริง ๆ เท่าใดนัก และนี่เองที่ช่วยให้การเขียนโปรแกรมภาษาแอสเซมบลีเป็นไปอย่างสะดวกมากขึ้น รวมทั้งช่วยสื่อความหมายได้ดีกว่าด้วย

เลเบลจะประกอบด้วยตัวอักษรต่าง ๆ ไม่เกิน 8 ตัว โดยจะเริ่มเลเบลที่คอลัมน์แรกของประโยคคำสั่งเท่านั้น เพราะถ้าหากเราเว้นช่องว่างไว้ในคอลัมน์แรกแล้ว ลิซ่าจะถือว่าประโยคนี้ไม่มีเลเบล และตัวอักษรที่อยู่ถัดไปจะอยู่ในส่วนที่สองหรือส่วนของนีโมนิคเลย

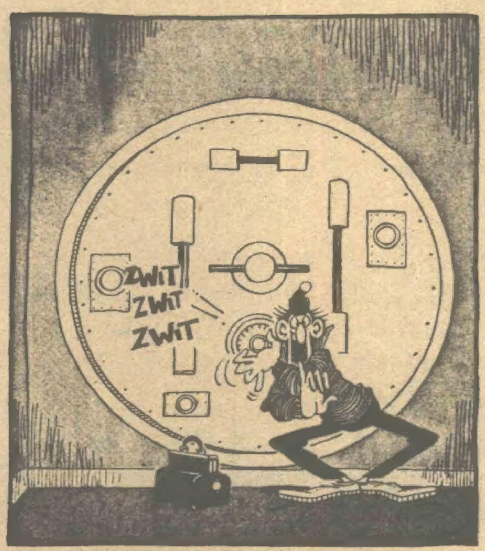
ข้อความที่จะใช้เป็นเลเบลนั้นเป็นข้อความอะไรก็ได้ แต่จะต้องไม่มีเครื่องหมาย ; หรือเว้นช่องว่างในเลเบลนั้น เพราะลิซ่าถือว่า ; เป็นเครื่องหมายที่ใช้นำหน้าส่วน COMMENT ซึ่งลิซ่าจะไม่สนใจอะไรกับ COMMENT เลย นอกจากนี้แล้ว ลิซ่ายังถือการใช้ช่องว่างในการแยกส่วนประกอบต่าง ๆ ออกจากกันด้วย ดังนั้นการเว้นช่องว่างในเลเบลก็อาจทำให้ลิซ่าเกิดความสับสนขึ้นได้

นีโมนิค เป็นส่วนที่ใช้แสดงการกระทำของประโยคคำสั่งนั้น ๆ นีโมนิคเป็นรหัสที่ใช้แทนรหัสภาษาเครื่องของซีพียูแต่ละเบอร์ เพราะคนเราคงไม่เข้าใจรหัสภาษาเครื่องที่เป็นตัวเลขฐานสิบหกเท่าใดนัก ดังนั้นจึงต้องหารหัสตัวอักษรมาใช้แทนรหัสตัวเลข เพื่อให้เราสามารถเข้าใจและสามารถนำไปใช้ได้สะดวก นี่แหละเป็นที่มาของรหัสนีโมนิคหล่ะ

สำหรับนีโมนิคของซีพียู 6502 นี้จะมีลักษณะเป็นตัวอักษรสามตัว เช่น LDA หรือ STX เป็นต้น นอกจากนีโมนิคในชุดคำสั่งของ 6502 แล้ว ลิซ่ายังสามารถเข้าใจรหัสอีกอย่างหนึ่งที่เรียกว่า คำสั่งเทียม ด้วย ตัวคำสั่งเทียมนี้ก็จะประกอบด้วยตัวอักษรสามตัวเช่นกัน เราไม่สามารถเปลี่ยนคำสั่งเทียมเป็นรหัสภาษาเครื่องได้ จะมีก็แต่ลิซ่าเท่านั้นที่สามารถเข้าใจและทำตามคำสั่งเทียมได้ สำหรับเรื่องของคำสั่งเทียมจะขอยกไปกล่าวให้ละเอียดอีกครั้งหนึ่งในตอนหลังนะครับ

โอเปอร์แรนด์ โดยทั่วไปเราจะใช้โอเปอร์แรนด์ร่วมกับนีโมนิค เพราะเมื่อมีการกระทำเกิดขึ้น โดยนีโมนิค ก็จะต้องบ่งตำแหน่ง หรือข้อมูล หรือที่เก็บข้อมูลที่มีส่วนร่วมในการกระทำนั้น ๆ เช่น ประโยคคำสั่ง JMP START มีส่วนนีโมนิค JMP ที่บอกให้รู้ว่ามีการกระโดดเกิดขึ้น แต่จะกระโดดไปยังที่ใดนั้น สามารถทราบได้โดยดูจากส่วนโอ–เปอร์แรนด์ ซึ่งในตัวอย่างนี้ก็คือให้กระโดดไปยังโปรแกรมที่มีเลเบล START นำหน้า

คำอธิบาย ถ้าจะเปรียบเทียบกับภาษาเบสิกแล้ว ส่วนนี้ก็คงจะเป็นส่วนที่ท้ายคำสั่ง REM กระมัง เพราะหน้าที่ของ COMMENT ก็คือเป็นคำอธิบายให้ผู้อ่านโปรแกรมสามารถทราบได้ง่ายว่าส่วนนี้เอาไว้ทำอะไร มีรายละเอียดอย่างไร เป็นต้น ส่วนของคำอธิบายจะต้องนำหน้าด้วยเครื่องหมาย ; เสมอ และจะต้องแยกจากส่วนอื่น ๆ โดยการใช้ช่องว่างด้วย



ส่วนที่อยู่ท้ายเครื่องหมาย ; นั้น เราจะใช้เป็นที่ว่างเขียนคำอธิบาย ซึ่งตัวลิซ่าจะไม่สนใจแปลคำอธิบายเหล่านี้เลย ดังนั้นเราจึงจัดส่วนของคอมเมนต์นี้ไว้ท้ายสุดของประโยคคำสั่ง

เอาล่ะ พอจะรู้จักกับส่วนต่าง ๆ ของภาษาแอสเซมบลีกันบ้างแล้ว เราก็ลองมาดูตัวอย่างโปรแกรมจริง ๆ กันสักหน่อย ซึ่งตัวอย่างในรูปที่ 6 นี้ จะแสดงให้เห็นส่วนต่าง ๆ ที่กล่าวมาแล้ว

ทีแรกนั้น ผมก็กะเอาไว้ว่าจะขอกล่าวถึงเรื่องชุดคำสั่งของ 6502 และเรื่องการอ้างแอดเดรสแบบต่าง ๆ สักหน่อย แต่เมื่อทราบว่ามีผู้เขียนเขียนถึงสองเรื่องนี้โดยละเอียดแล้ว ผมเองก็ไม่อยากจะเขียนซ้ำอีก (กลัวจะอธิบายได้ไม่ดีเท่าเขา) จึงขอข้ามมาพูดถึงเรื่องการกำหนดค่าตัวเลขในภาษาแอสเซมบลีกันซะเลย เนื่องมาจากว่าภาษาแอสเซมบลีนั้นมีส่วนเกี่ยวข้องกับเรื่องของตัวเลขอย่างมาก ไม่ว่าจะเป็นค่าของแอดเดรสหรือข้อมูลต่าง ๆ ก็เป็นตัวเลขเหมือนกันหมด ดังนั้นเรื่องของการกำหนดค่าตัวเลขในโปรแกรมแอสเซมบลี จึงเป็นเรื่องสำคัญเรื่องหนึ่ง

ลิซ่าเป็นโปรแกรมแอสเซมเบลอร์ตัวหนึ่งที่มีความสามารถด้านตัวเลขมาก สามารถยอมรับตัวเลขฐานต่าง ๆ ได้ด้วยกันสามฐานคือ เลขฐานสอง, ฐานสิบ และฐานสิบหก นอกจากนี้ยังมีความสามารถเข้าใจ และคำนวณค่าตัวเลขได้จากสมการที่มีตัวแปรเป็นเลเบล และเครื่องหมายการกระทำทางคณิตศาสตร์หรือทางลอจิกได้ด้วย

หลักเกณฑ์ที่ใช้กำหนดค่าตัวเลขได้แก่

1. เลขฐานต่าง ๆ จำเป็นจะต้องมีเครื่องหมายพิเศษนำหน้าตัวเลขเหล่านั้น และตัวเลขนั้นจะต้องมีค่าไม่เกินไปกว่าขอบเขตที่กำหนดดังนี้

เลขฐานสิบหก ใช้เครื่องหมาย $ นำหน้าตัวเลขตั้งแต่ $0000-$FFFF

เลขฐานสิบ ใช้เครื่องหมาย ! นำหน้าตัวเลขตั้งแต่ 0-65535

เลขฐานสอง ใช้เครื่องหมาย % นำหน้าตัวเลขตั้งแต่ %0000 0000 0000 0000-%1111 1111 1111 1111

2. สามารถใช้เครื่องหมายการกระทำทางคณิตศาสตร์ และทางลอจิกประกอบการกำหนดค่าตัวเลขได้ด้วย โดย

ใช้ +, −, *, / แทนเครื่องหมายบวก, ลบ, คูณ และหาร ตามลำดับ

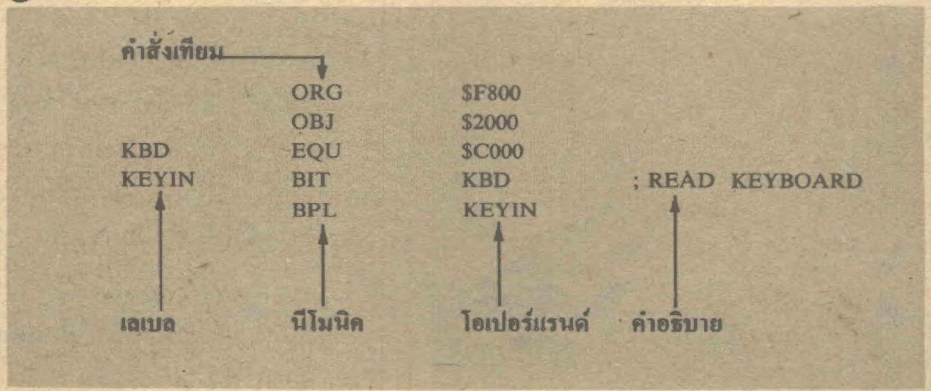
ใช้ & แทนการแอนด์ลอจิก (AND)

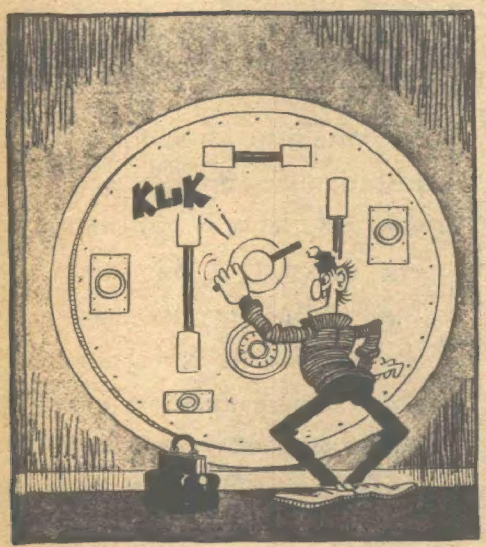
ใช้ I แทนการ OR (โดยการกดปุ่ม ESC และปุ่ม I ตามลำดับ)

ใช้ ∧ แทนการ EXCLUSIVE OR

3. สามารถกำหนดค่าตัวเลขขนาด 8 บิท จากค่าตัวเลขขนาด 16 บิทได้ โดยใช้เครื่องหมาย # เมื่อต้องการตัวเลขไบท์ต่ำ และใช้เครื่องหมาย / เมื่อต้องการตัวเลขไบท์สูง ดังเช่น กำหนดให้ START

6





= $FA10 ดังนั้น#START จะมีค่าเท่ากับ $10 และ /START มีค่าเท่ากับ $FA เป็นต้น

4. ในกรณีที่มีการกำหนดค่าเป็นสมการ ลิซ่าจะเริ่มทำการคำนวณค่าจากทางซ้ายมาหาทางขวา เช่น $5-$3-$2 = $0 เป็นต้น

5. ลิซ่าสามารถเปลี่ยนตัวอักษรต่าง ๆ ไปเป็นรหัสตัวเลขที่เรียกว่า รหัสแอสกี ได้ด้วย โดยใช้เครื่องหมาย ' หรือ " แต่มีข้อแตกต่างกันเล็กน้อยคือ เมื่อใช้เครื่องหมาย ' จะมีการเซทบิทที่ 7 ของรหัสแอสกีให้มีค่าเป็น 0 ตรงกันข้ามถ้าหากใช้เครื่องหมาย " บิทที่ 7 ของรหัสแอสกีจะกลายเป็น 1 แทน เช่น

'B' มีค่าเท่ากับ $42 และ "B" มีค่าเท่ากับ $C2 เป็นต้น

นอกจากนี้ เรายังสามารถใช้ 'B แทนการใช้#'B'(หรือ "B แทนการใช้ #"B") ในการอ้างแอดเดรสแบบ IMMEDIATE MODE เช่น 'B' เป็นการใส่ค่า $42 ลงในรีจิสเตอร์ A เราสามารถเขียนใหม่ได้ดังนี้ LDA 'B เป็นต้น

คำสั่งเทียม

ยังจำได้ไหมว่า ผมเคยเกริ่นถึงเรื่องคำสั่งเทียมไปนิดหน่อยแล้ว มาถึงตอนนี้ก็จะขอกล่าวถึงรายละเอียดกันเสียเลยนะ

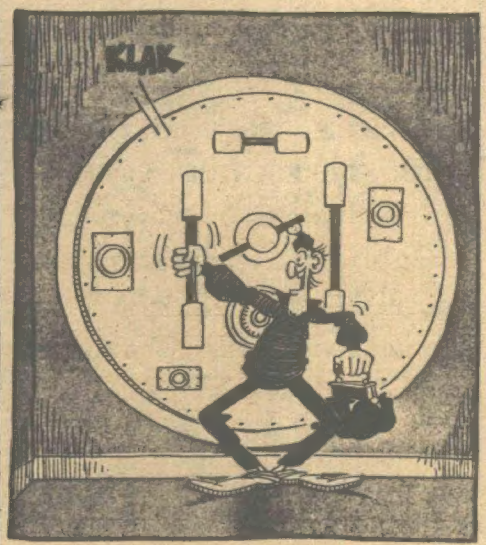
คำสั่งเทียมที่ลิซ่าสามารถเข้าใจได้มีหลายคำสั่งด้วยกัน แต่ผมได้เลือกเอาเฉพาะที่ใช้กันบ่อย ๆ มาอธิบายให้ฟังเท่านั้น ชุดคำสั่งเทียมเหล่านั้นแสดงไว้ในรูปที่ 7

เราลองมาดูรายละเอียดของแต่ละคำสั่งกันสักหน่อยนะครับ โดยเริ่มจาก...

OBJ(OBJECT CODE ADDRESS) เมื่อเราสั่งให้ลิซ่าทำการเปลี่ยนโปรแกรมภาษาแอสเซมบลี ไปเป็นโปรแกรมภาษาเครื่องหรือออบเจคโค้ดที่คุณต้องการนั้น ลิซ่าก็จะทำตามที่คุณสั่ง แต่คุณรู้ไหมว่าออบเจคโค้ดที่ได้นั้นถูกเก็บไว้ที่ไหนในหน่วยความจำ ถ้าหากคุณไม่กำหนดตำแหน่งที่เก็บลงไปในโปรแกรมแอสเซมบลีแล้ว ลิซ่าก็จะนำออบเจคโค้ดไปเก็บไว้ในบริเวณที่มันจัดไว้โดยเฉพาะ เป็นเนื้อที่ขนาด 4 K ตั้งแต่ $800-$1800 แต่ถ้าหากคุณต้องการเปลี่ยนตำแหน่งที่เก็บโปรแกรมภาษาเครื่องก็ขอให้ใช้คำสั่งเทียม OBJ นี้ กำหนดค่าแอดเดรสไว้หน้าโปรแกรมแอสเซมบลี แต่ควรจะให้อยู่ในขอบเขตที่ลิซ่าจัดไว้เท่านั้น คือ ตั้งแต่ $800-$1800

ORG(PROGRAM ORIGIN) เป็นคำสั่งเทียมที่ใช้กำหนดค่าแอดเดรสเริ่มต้นของโปรแกรม การอ้างอิงค่าแอดเดรสต่าง ๆ จะถือแอดเดรสที่กำหนดโดยคำสั่ง ORG เป็นหลัก ถ้าหากคุณไม่กำหนดคำสั่ง ORG ไว้ในโปรแกรม ลิซ่าก็จะถือว่าโปรแกรมนี้เริ่มต้นที่แอดเดรส $800

คำสั่ง ORG นั้นนอกจากจะกำหนดแอดเดรสเริ่มต้นของโปรแกรมแล้ว ยังกำหนดตำแหน่งที่เก็บของออบเจคโค้ดไปในตัวด้วย เช่น ถ้าหากคุณใช้คำสั่ง ORG $1000 คุณก็จะได้ออบเจคโค้ดเก็บไว้ที่แอดเดรส $1000 ด้วย คุณสามารถจะรันโปรแกรมนั้นโดยใช้คำสั่ง 1000G ได้เลย แต่ถ้าหากคุณต้องการเริ่มต้นโปรแกรมที่แอดเดรส $4000 ตัวออบเจคโค้ดจะเก็บไว้ที่แอดเดรส $4000 ไม่ได้ เพราะไม่ใช่ขอบเขตที่ลิซ่ากำหนด ดังนั้น



7

OBJ	ORG	EPZ	EQU	ASC
STR	DCI	INV	BLK	HEX
ADR	DBY	HBY	BYT	DFS
LST	NLS	DCM	END	

คุณจึงจำเป็นต้องใช้คำสั่ง OBJ เพื่อกำหนดที่เก็บออบเจคโค้ดที่ถูกต้องดังนี้

ORG $4000 ด้วยคำสั่งเทียม 2 บรรทัดนี้ ออบเจคโค้ดก็จะถูกเก็บไว้ที่แอดเดรส $800

OBJ $800 ถ้าหากคุณต้องการรันโปรแกรมนี้ ก็จำเป็นจะต้องย้ายโปรแกรมจาก $800 ให้ไปเริ่มต้นที่แอดเดรส $4000 เสียก่อน จึงจะรันโปรแกรมนั้นได้

EPZ(EQUATE TO PAGE ZERO) เป็นคำสั่งกำหนดค่าตัวแปร เลเบลต่าง ๆ สำหรับการอ้างแอดเดรสไปที่หน่วยความจำเพจศูนย์ เนื่องจากเป็นการกำหนดค่าแอดเดรสในเพจศูนย์ ดังนั้นค่าตัวเลขจึงอยู่ในช่วง $00-$FF เท่านั้น การใช้คำสั่งเทียม EPZ จะอยู่ในรูปแบบดังนี้

เลเบล EPZ ค่าแอดเดรส

เช่น PCH EPZ $34 หรือ PCL EPZ PCH-$1 เป็นต้น

EQU(EQUATE) เป็นคำสั่งเทียมที่ใช้กำหนดค่าของเลเบลทั่ว ๆ ไป โดยเป็นค่าตัวเลขขนาด 16 บิท ตั้งแต่ $0000-$FFFF คุณอาจใช้เครื่องหมาย = แทนคำสั่ง EQU ก็ได้

ASC(ASCII STRING DEFINITION) ใช้สำหรับเปลี่ยนตัวอักษรต่าง ๆ (STRING) ให้อยู่ในรหัสแอสกีดังตัวอย่างการใช้งานดังนี้

ASC "HELLO" จะได้เป็น C8 C5 CC CC CF

ASC'HELLO' จะได้เป็น 48 45 4C 4C 4F

จะเห็นว่าเครื่องหมาย " และ ' ยังคงมีผลต่อบิทที่ 7 ของรหัสแอสกีอยู่เหมือนเดิมดังที่กล่าวมาแล้ว

STR(CHARACTOR STRING DEFINITION) มีลักษณะการใช้งานคล้าย ๆ กับคำสั่ง ASC คือใช้เปลี่ยนตัวอักษรต่าง ๆ ให้อยู่ในรูปของรหัสแอสกีเช่นกัน แต่คำสั่ง STR นั้นมีข้อพิเศษกว่าคำสั่ง ASC ตรงที่สามารถให้ค่าความยาวของตัวอักษรชุดนั้น ๆ ออกมาด้วย เช่น

STR "HELLO" จะได้เป็น 05 C8 C5 CC CC CF

05 นั้นเป็นค่าความยาวของสตริง HELLO

DCI(DEFINE CHARACTOR IMMEDIATE) เป็นอีกคำสั่งหนึ่งที่เกี่ยวข้องกับ STRING และมีผลลัพธ์คล้ายกับ ASC จะต่างกันตรงรหัสแอสกีของตัวอักษรตัวสุดท้ายของสตริง ซึ่งจะเปรียบเทียบให้ดูความแตกต่างดังนี้

ASC "ABC" ได้เอาท์พุทเป็น C1 C2 C3

DCI "ABC" ได้เอาท์พุทเป็น C1 C2 (43)←บิท 7 ของแอสกีเป็นศูนย์

DCI 'ABC' ได้เอาท์พุทเป็น 41 42 (C3)←บิท 7 ของแอสกีเป็นหนึ่ง

INV(INVERTED CHARACTER) และ BLK(BLINKING CHARACTER) เป็นคำสั่งที่ให้รหัสตัวอักษรในการแสดงผลบนจอภาพในลักษณะอินเวอร์สและกะพริบ เช่น

INV 'ABC' จะให้ผลเป็น 01 02 03

BLK 'ABC' จะให้ผลเป็น 41 42 43 เป็นต้น

HEX (HEXADECIMAL STRING DEFINITION) เป็นคำสั่งสำหรับเก็บค่าคงที่ ซึ่งเป็นตัวเลขฐานสิบหกไว้ในโปรแกรม โดยค่าตัวเลขที่อยู่หลังคำสั่งนี้ ไม่จำเป็นจะต้องนำหน้าด้วยเครื่องหมาย $ และจะประกอบด้วยตัวเลขกี่ตัวก็ได้ แต่จะต้องเขียนติด ๆ กัน เช่น

HEX FF0307AB00 เป็นต้น

ADR(ADDRESS STORAGE) ด้วยคำสั่งเทียมนี้ คุณก็จะสามารถเก็บ

ค่าตัวเลขขนาด 16 บิทไว้ในโปรแกรม ได้โดยอยู่ในรูปของตัวเลข 8 บิทสองตัว และตัวเลขไบท์ต่ำจะถูกเก็บไว้ข้างหน้าตัวเลขไบท์สูงตัวเลขที่กำหนดอยู่หลังคำสั่ง ADR สามารถกำหนดเป็นค่าคงที่หรือกำหนดเป็นชื่อเลเบลด้วยก็ได้ เช่น

ADR $FB08 จะได้เป็น 08 FB

ADR LB1, LB2, LB3

เราสามารถใช้คำสั่ง ADR ในการเก็บข้อมูลขนาด 16 บิท หรือสร้างตารางกระโดด (JUMP TABLE) ไว้ในโปรแกรมก็ได้ นอกจากนี้เรายังสามารถใช้ ADR ในการเก็บค่าคงที่ที่เป็นตัวเลขฐานอื่น ๆ ได้ด้วย เช่น

ADR ! 4096, %1010

DBY(DOUBLE BYTE DATA) เป็นคำสั่งที่มีลักษณะเกือบทุกอย่างคล้ายคลึงกับคำสั่ง ADR จะต่างกันก็ตรงที่คำสั่ง DBY จะเรียงตัวเลขไบท์สูงไว้หน้าตัวเลขไบท์ต่ำ ในขณะที่คำสั่ง ADR จะวางตัวเลขไบท์ต่ำนำหน้าตัวเลขไบท์สูง ดังตัวอย่าง

ADR $ 1020 จะได้เป็น $20 $10

DBY $1020 จะได้เป็น $10 $20

HBY(HIGH BYTE DATA) และ BYT(LOW BYTE DATA) ใช้สำหรับเก็บค่าตัวเลขขนาด 8 บิท จากตัวเลขขนาด 16 บิท โดยคำสั่ง HBY จะตัดเอาเฉพาะตัวเลขไบท์สูงไปเก็บไว้ ในขณะที่คำสั่ง BYT จะเก็บเฉพาะตัวเลขไบท์ต่ำเท่านั้น เมื่อทราบถึงหน้าที่ของสองคำสั่งนี้แล้ว ก็ทำให้อดนึกถึงเครื่องหมาย #และ / ที่กล่าวมาแล้ว คงยังจำกันได้ว่าเราใช้ 2 เครื่องหมายนี้ในการกำหนดค่าตัวเลขขนาด 8 บิทจากตัวเลขขนาด 16 บิท แต่จะกำหนดค่าตัวเลขได้เพียงทีละไบท์เท่านั้น ส่วนคำสั่ง HBY และ BYT นั้นสามารถใช้เก็บค่าตัวเลขได้ทีละหลาย ๆ ไบท์เลย เช่น

HBY $1234, "A" จะได้เป็น $12 $00

DFS(DEFINE STORAGE) เราใช้คำสั่งเทียม DFS ในการจองเนื้อที่ของหน่วยความจำ มีลักษณะการใช้งานดังนี้ เลเบล DFS N, M เป็นการจองเนื้อที่ขนาด N ไบท์ ไว้ที่ตำแหน่งที่กำหนดด้วยเลเบลนั้น แล้วให้ใส่ค่าตัวเลข M ลงในหน่วยความจำที่จองเอาไว้นั้น (ค่าตัวเลข M อาจจะกำหนดไว้หรือไม่มีเลยก็ได้)

LST(LISTING OPTION ON) และ NLS(NO LISTING) สองคำสั่งนี้จัดว่าเป็นคำสั่งคู่กัน โดยคำสั่ง LST จะเป็นเหมือนคำสั่งเปิดสวิทช์ให้ลิซ่าแสดงโปรแกรมและออบเจคโค้ดออกมา ขณะที่ลิซ่าทำการแปลภาษาด้วย แต่ถ้าไม่ต้องการให้ลิซ่าแสดงอะไรออกมาเลยขณะทำการแปลภาษา ก็สามารถใช้คำสั่ง NLS สั่งงานลิซ่าได้ บางครั้งเราอาจจะใช้คำสั่ง LST และ NLS ร่วมกันเพื่อแสดงเฉพาะบางส่วนของโปรแกรมก็ได้

DCM(DISK COMMAND) ในขณะที่ลิซ่ากำลังทำการแอสแซมเบิ้ลโปรแกรมอยู่นั้น เราสามารถจะสั่งงานให้ลิซ่าทำงานตามคำสั่งในดอสที่เรากำหนดไว้ได้ โดยวางคำสั่งนั้นอยู่หลังคำสั่งเทียม DCM เช่นเดียวกับที่เราใช้ปุ่ม ctrl-D นำหน้าคำสั่งในดอสดังที่กล่าวมาแล้วในเรื่องคำสั่งของลิซ่า

เราสามารถใช้คำสั่ง PR#N หรือ IN#N รวมกับคำสั่งเทียม DCM ได้ด้วย ดังตัวอย่าง

DCM "PR#1" หรือ DCM "BSAVE MUSIC, AS 800, LS 100" เป็นต้น

นอกจากนี้ หากคุณทำการแอสเซมบลีโปรแกรมที่มีความยาวมาก ๆ จนขนาดของออบเจคโค้ดใหญ่เกินเลยไปจากขนาดหน่วยความจำ 4 K ที่ลิซ่ากำหนดไว้ให้เป็นที่เก็บออบเจคโค้ดโดยเฉพาะ ดังนั้นเราจำเป็นต้องทำการแปลโปรแกรมทีละ 4 K แล้วจัดการเก็บโปรแกรมภาษาเครื่องนั้นลงไปในแผ่นดิสก์ก่อน แล้วจึงทำการแปลโปรแกรมส่วนต่อไป ด้วยวิธีนี้ เราจึงต้องใช้คำสั่งเทียม DCM และ OBJ ร่วมกันดังในรูปที่ 8

เมื่อเวลาใช้งานโปรแกรมนี้ ก็ให้นำโปรแกรมมาเรียงต่อกันทีละส่วนจนครบ ก็จะสามารถรันโปรแกรมได้แล้ว

END(END OF ASSEMBLY) คำสั่ง END จะถูกใช้เพื่อบอกให้ลิซ่าทราบว่าจบโปรแกรมแอสเซมบลีแล้ว เมื่อลิซ่าพบกับคำสั่ง END แล้ว ลิซ่าก็จะโดดออกจากแอสเซมเบิ้ลโหมดเลย ดังนั้นคำสั่ง END จึงควรวางไว้ท้ายสุดของโปรแกรม แต่ถ้าหากคุณลืมใส่คำสั่ง END ไว้ในโปรแกรม ลิซ่าก็จะฟ้องว่าเกิดข้อผิดพลาดขึ้น ให้คุณเติมคำสั่ง END ลงไปซะ

เมื่อคุณอ่านบทความมาจนถึงตอนนี้แล้ว คุณก็สามารถใช้งาน LISA ในการเขียนโปรแกรมแอสเซมบลีได้แล้ว รวมทั้งสามารถใช้คำสั่งเทียมร่วมกับคำสั่งจริงในโปรแกรมของคุณได้ด้วย โอกาสหน้าจะเสนอการใช้ลิซ่าเขียนโปรแกรมให้มีประสิทธิภาพมากยิ่งขึ้น

8

```
OBJ $800
โปรแกรมขนาด 4 K
DCM "BSAVE PARTONE, A$800, L$1000"
OBJ $800
โปรแกรมขนาด 4K
DCM "BSAVE PARTTWO, A$800, L$1000"
OBJ $800
```